Python - zadaci za vežbu

1. Tipovi i objekti

Zadatak 1: Prekucati prvi primer programa. Umesto print(year, principal) staviti da se ispiše: U godini *godina* principal je *principal* upotrebom format funkcije. Principal ispisati zaokruženo na jednu decimalu. Ispisati tip promenljive *principal* pre i posle petlje koristeći type(principal)

Zadatak 2: Napraviti dve int promenljive jednake vrednosti. Proveriti jednakost dva objekta (preko is) i jednakost po vrednosti (preko ==). Zatim napraviti dve jednake liste, koje sadrže neke int vrednosti. Recimo, [1, 2, 3, 4]. Ispitati jednakosti.

Zadatak 3: Napraviti listu sa matematičkim operatorima add, sub i mul (add(x,y) je isto kao x + y. sub x-y, mul x*y). Napraviti sekvencu od dva broja. Primentiti sve operacije nad ta dva broja ispisujući rezultat. Importovati operatore pomoću: `from operator import add, mul, sub`

Zadatak 4: Napraviti listu nekih reči. Na osnovu nje kreirati listu čiji n-ti član odgvara dužini n-te reči prve liste.

Zadatak 5: Zadati neki string i odrediti frekvenciju pojave svakog karaktera u njemu. Frekvenciju predstaviti rečnikom.

Zadatak 6: Simulirati datu switch naredbu pomoću rečnika i njegove get metode sa default-nom vrednošću.

```
String string;

switch(x){
case 10:
	string = "deset";
	break;
case 9:
	string = "devet";
	break;
case 8:
	string = "osam";
	break;
case 7:
	string = "sedam";
	break;
case 6:
	string = "sest";
	break;
default:
	string = "nije polozeno";
	break;

}
System.out.println(string);
```

2. Struktura programa i kontrola toka

Zadatak 1: U zavisnosti od vrednosti neke Boolean varijable, definisati funkciju *operacija(a,b)* kao sabiranje prosleđenih brojeva u jednoj varijanti i oduzimanje u drugoj. Pozvati funkciju i ispisati rezultat.

Zadatak 2: U engleskom jeziku present participle se pravi dodavanje sufiksa -ing na infinitivnu formu. Mogu se definisiati sledeća heuristička pravila (jednostavnosti radi, ne treba pokriti njihove izuzetke):

1. Ako se glagol završava na ie, promeniti ie u y i dodati ing
2. Ako se glagol završava sa samo na e, izbaciti ga i dodati ing
3. Ako je reč oblika suglasnik - samoglasnik - suglasnik, duplirati poslednje slovo pre dodavanja ing
4. Inače, samo dodati ing

Definisati funkciju *make_ing_form*(*verb)* koja transformise glagol u infinitivu u present participle. Proveriti za *lie, move, run, stand.* Koristiti isecanje (*slicing*). Prvo proveriti da li su svi karakteri slova.

Zadatak 3: Napisati funkciju koja prima dve liste brojeva i vraća treću čiji je svaki član jednak zbiru članova prve i druge liste sa istim indeksom. Ako dve liste nisu iste dužine, vratiti None. Napisati dve varijane funkcije. U prvoj koristiti *zip* za iteraciju. U drugoj interirati po indeksima (for petlja od prvog do poslednjeg indeksa - koristiti *range*).

3. Funkcije i funkcionalno programiranje

Zadatak 1: Definisati rečnik koji će se popunjavati pravoima - id ranika, lista jezika koje zna na dovoljno visokom nivou. Zatim, napisati funkciju koja prima id radnika, potrebni nivo poznavanja jezika i jezike koje poznaje sledećeg zaglavlja: *languages(employee_id, level, **languages).* Imenovane parametre proslediti tako da naziv parametra bude naziv jezika, a vrednost nivo na kom ga radnik zna (neki broj od 1 do 5). U funkciji kreirati listu sa jezicima koje radnik zna na nivou bar jednakom vrednosti prosleđenoj kao *level*, a zatim je dodati u definisani rečnik.

Primer poziva: languages(1, 3, English = 5, German = 3)

Zadatak 2: Napisati funkciju koja prima listu reči i vraća najdužu i njen indeks. Koristiti enumerate. Rezultat vratiti u vidu torke.

Zadatak 3: Napisati dekorator funkciju koja ispisuje vreme i naziv funkcije koja se poziva, poziva datu funkciju, i ispisuje vreme završteka, naziv i rezultat funkcije. Trenutno vreme se dobija na sledeći način:

```
import datetime
datetime.datetime.now()
```

Zadatak 4: Pomoću list comprehension mehanizma napraviti:

1. Listu koja sadrži sve reči druge liste duže od 3
2. Listu kvadrata svih parnih brojeva između 1 i 10
3. Rešiti 4. zadatak iz Tipova i objekata u jednoj liniji pomoću list comprehension-a

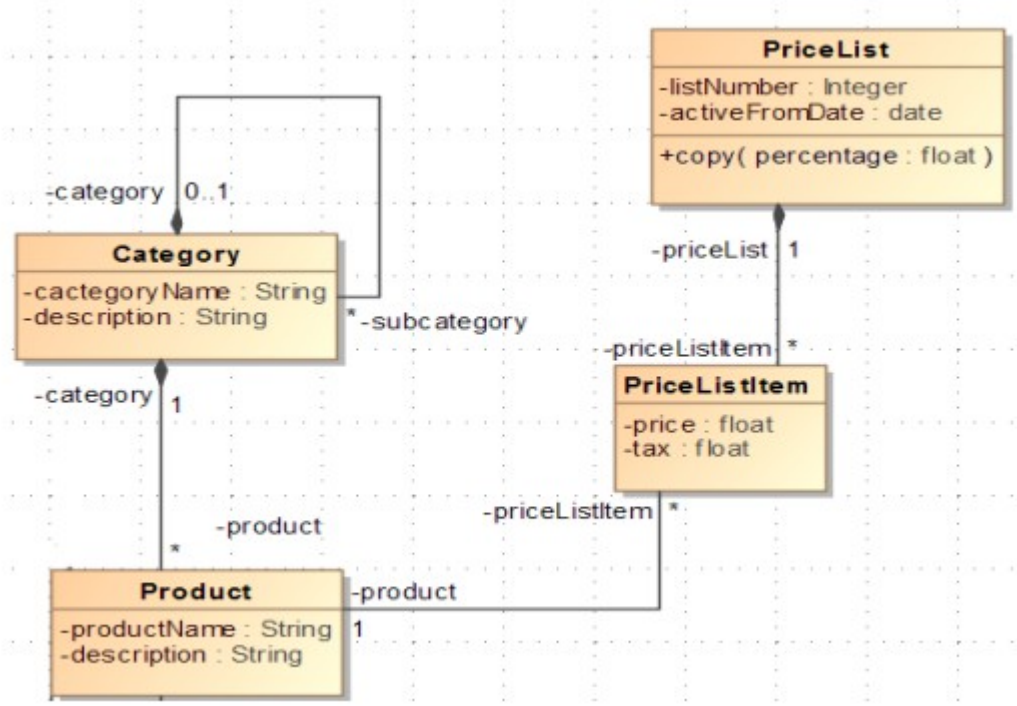
4. Listu koja sadrži reči obrnute reči druge list ('primer' – 'remirp'). Korisititi *slicing* za obrtanje reči

Zadatak 5: Napraviti generator izraz koji izračunava kubove svih neparnih brojeva između 1 i 1000. Ispisati prvih nekoliko vrednosti. Napomena: U Pyton-u 3 se koristi *__next__()* umesto *next()*. Za stepenovanje koristiti operator **.

Zadatak 6: Napisati i pozvati anonimnu funkciju koja prima dva broja kao argumente i računa njihov proizvod.

4. Klase i objektno-orijentisano programiranje

Zadatak 1: Implementirati deo modela web shop-a sa slike:



Napraviti jedan PriceList objekat sa nekoliko stavki, ispisati je. Napraviti novu listu kopiranjem date putem copy() metode.

Zadatak 2: Objekti mogu implementirati jedan ili više relacionih operatora (<,>,<=,>=,==,!=). Potrebno je samo implementirati specijalne metode za poređenje, koje primaju dva argumenta i vraćaju bilo koji Python tip kao rezultat, što naravno može biti i Boolean. U pitanju su sledeće metode (objekat ih ne mora sve implementirati, nego samo neke od njih, po potrebi):

__lt__(self,other) - self < other
__le__(self,other) - self <= other
__gt__(self,other) - self > other

__ge__(self,other) - self >= other
__eq__(self,other) - self == other
__ne__(self,other) - self != other

Implementirati metodu(e) koje su neophodne da bi se mogla sortirati lista price list item-a od stavke sa najmanjom cenom do stavke sa najvećom, te pozvati spomenuto sortiranje.

Zadatak 3: Napisati klasu sa dva privatna atributa *x* i *y* i omogućiti da im se pristupa i menja vrednost izvan klase. Pri postavljanju vrednosti baciti grešku ako vrednosti nije int. Koristiti *property* i *setter,* ne pisati klasične getter-e i setter-e!

5. Moduli i paketi

Zadatak 1: Napraviti dva paketa. U prvi staviti neki modul sa nekim funkcijama i/ili klasama. U okviru tog modula staviti da se nešto ispisuje, bilo pozivanjem neke njegove funkcije, bilo prostim dodavanjem neke print funkcije direktno u modul. U drugom paketu napraviti modul u okviru kog se importuje modul iz prvog paketa:

1. Pomoću import naredbe
2. Pomoću from naredbe

Nakon importa, pozvati neku funkciju ili instancirati klas iz importovanog modula.

Napomena: modul unutar paketa se imenuje pomoću imena paketa: paket.modul. Ako se koristi PyDev, i tako se naznači pri kreiranju projekta, src folder se automatski doda na PYTHONPATH. PYTHONPATH govori interpreteru gde da gleda kada traži modul za importovanje. Dodavanjem src foldera na ovu varijablu izbegava se potreba za relativnim referenciranjem. Ručno možemo dodati neki direktorijum na PYTHONPATH pomoću `sys.path.append(dir)`

Zadatak 2: Da bi se omogućilo importovanje svih modula iz nekog paketa, nije dovoljno samo pozvati `from paket import *`. Neophodno je prvo definisati u __init__.py unutar paketa spisak svih njegovih sadržanih modula koji se importuju na sledeći način:

```
__all__ = ["modul1","modul2","modul3"]
```

Importovati modul prvog paketa iz prethodnog zadatke preko import *

6. Ulaz i izlaz

Zadatak 1:

1. Python interpreter pruža tri standardna fajl objekta za čitanje i pisanje - *standard input, standard output i standard error*, koji su dostupni preko *sys* modula - *sys.stdin, sys.stdout i sys.stderr.* Najčešće se koriste za ćitanje unosa sa tastature i ispis na ekran. Koristeći ove objekte, pitati korisnika da unese ime, učitati ga i ispisati.
2. Alternativa za učitavanje je korišćenje ugrađene funkcije *input([prompt]),* gde je opcioni argument tekst koji se ispisuje pre nego što se korisnik dobije mogućnost unosa odgovora. Realizovati istu komunikaciju sa korisnikom (traženje i ispis imena) pomoću ove funkcije.

Zadatak 2: Za operacije vezane za operativni sistem (putanje, kreiranje foldera) i sl. koristi se *os*

modul. Česta praksa da se dođe do putanje korena projekta je da se upravo tamo stavi jedan modul, te se iščita putanja do njegovog direktorijuma. To se postiže na sledeći način:

```
import os

root_dir = os.path.abspath(os.path.dirname(rootmodule.__file__))
```

A ako je modul direktno unutar src foldera (pa nam treba jedan folder pre):

```
root_dir = os.path.split(os.path.abspath(os.path.dirname(rootmodule.__file__)))[0]
```

Dalje možemo spojiti tu putanju sa relativnom putanjom do resursa koji nam treba na sledeći način:

```
path = os.path.join(root_dir, [“folder1”, “folder2”,…], “file”)
```

Napraviti folder resources u korenu projekta, a u njemu jedan tekstualni fajl. Iščitati i ispisati sadržaj tog fajla. Obuhvatiti ovaj kod u “try - except“ blok.

Napomena [i] označavaju da su parametri opcioni, ne da se treba proslediti lista. Isto važi i za sledeći zadatak.

Zadatak 3: Napisati funkciju *char_freq_table()* koja od korisnika traži da unese naziv nekog tekstualnog fajla (pretpostaviti da se falj nalazi u kreiranom folderu resources). Ako fajl ne postoji, prijaviti grešku korisniku. Ako postoji, broji koliko ima pojava svakog slova u tekstu ne razlikujući mala od velikih (preskočiti ostale karaktere) i rezultat upisuje u neki drugi fajl u resources folderu u obliku:

karakter 1: broj pojava 1

karakter 2: broj pojava 2

itd.

Zadatak 4: Serijalizacija nekog objekta u fajl može se vršiti veoma jednostavno pomoću *pickle* modula:

```
f = open(filename,'wb')

pickle.dump(obj, f, [pickle.HIGHEST_PROTOCOL])

f.close()
```

te učitati iz fajla na sledeći način:

```
f = open(filename,'rb')

obj = pickle.load(f)

f.close()
```

Realizovati snimanje i učitavanje nekog objekta (instance neke napisane klase, neke liste...) u fajl i iz fajla unutar resources foldera.